Genesis: An Extensible Java

by

Ian Lewis BComp Hons

Submitted in fulfilment of the requirements for the

Degree of Doctor of Philosophy

University of Tasmania

February 2005

This thesis contains no material which has been accepted for a degree or diploma by the University or any other institution, except by way of background information and duly acknowledged in the thesis, and to the best of the candidate's knowledge and belief no material previously published or written by another person except where due acknowledgement is made in the text of the thesis.

Ian Lewis

Ian Lewis

Abstract

Extensible programming languages allow users to create fundamentally new syntax and translate this syntax into language primitives. The concept of compile-time meta-programming has been around for decades, but systems that provide such abilities generally disallow the creation of new syntactic forms, or have heavy restrictions on how, or where, this may be done.

Genesis is an extension to Java that supports compile-time meta-programming by allowing users to create their own arbitrary syntax. This is achieved through macros that operate on a mix of both concrete and abstract syntax, and produce abstract syntax. Genesis attempts to provide a minimal design whilst maintaining, and extending, the expressive power of other similar macro systems.

The core Genesis language definition lacks many of the desirable features found in other systems, such as quasi-quote, hygiene, and static expression-type dispatch, but is expressive enough to define these as syntax extensions. User-defined macros produce only well-formed syntactic structures via the use of a predefined set of classes that define a Java abstract syntax.

At the heart of Genesis is a flexible parser that is capable of parsing any context-free grammars — even ambiguous ones. The parser is capable of arbitrary speculation and will consider all possible parses. The parser constructs a graph of possible paths, and is capable of dynamically pruning this graph, or combining nodes, based on precedence or associativity rules. This general parser allows macro programmers to forget about parsing, and concentrate on defining new syntax.

One key goal of this system was to address the programmer's learning curve by providing as simple a system as possible. This was achieved by the use of the flexible parser, the introduction of only one new construct to standard Java, and extensions to make programming macros more user friendly.

The expressiveness of Genesis is wide ranging; it is capable of providing small scale limited use macros, large scale semantic modifications, through to complete language replacements.

To demonstrate this expressiveness, we implement many of the simple test cases found in other systems, such as a type-safe `printf`, assertions, and iteration statements. These test cases require an ability to perform static type-checking and to manipulate compile-time values and abstract syntax trees. As additional examples of Genesis' expressive power we also provide implementations of embedded subsets of SQL and Haskell. As a final proof of power, the Haskell subset can operate as a stand-alone extension independent of any recognisable Java code.

Acknowledgements

Thanks to Vishv for being my most long-suffering supervisor and for steadfastly maintaining to this day that he has no idea what I am doing whatsoever.

Also, to Arthur and Julian for helping get me over the line, battered and bruised though I may be.

Thanks to the School of Computing for providing support when it was really necessary and without which I would never even come close to finishing.

To Julian for showing me no matter how annoyingly pedantic I may have thought myself there is always someone worse, and who will scare me.

To Nicole simply for torturing countless cats (as is my understanding) rather than slaying me alive simply because, as my office neighbour, she was Ned to my Homer.

To Adam for giving us all someone to mutter incessantly under our breaths about at his appearance of progress and for all the tennis ball humour — just hilarious. Oh yeah, and for proof-reading.

Thanks to Mark for all of those closed door discussions, the whiteboard Jeffisms, misogynistic research proposals, and for becoming a living urban legend.

To Byeong for showing me that it really is worth taking the time to greet people, and for being brave enough to come drinking with me — once.

To Craig, for scheduling his holiday for the worst possible time… and for making up for it by scheduling another celebratory one once this ordeal is finally over.

Thanks to Banjo's, Hungry Jacks, Eastlands, Centrepoint, Northgate, KFC, and their ilk for donating places where, strangely, I initially found myself most productive. Particularly thanks to those who donated their power to make this all possible.

To my darling girlfriend Kim for tolerating all those assurances of "two more weeks", and just being tolerant in general.

And finalee, tanks to those harmleff alcamahol molecools that didn't hampa me won bit.

Table of Contents

List of Code Examples

List of Figures

List of Tables